CASTILLO, Diana Vanessa S.
CoE 197S ME8 Documentation

Exercise 6: Networking

1. Listing Networks

```
PS C:\Users\whatanicedayiana\Documents\CoE197S\ME8_Containerization_and_Docker\5-volumes> docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
3f805bbb1492   bridge    bridge    local
736823f9ad72   host      host      local
dcdc3c866b69   none      null      local
```

Fig. 1. List of networks

2. The default bridge network

```
PS C:\Users\whatanicedayiana\Documents\CoE197S\ME8_Containerization_and_Docker\5-volumes> docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "3f805bbb14928370c76ef1d254df6b1ac0cfb2623c4db3ced9571ec208371d6c",
        "Created": "2021-06-11T19:32:15.6929291Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
```

Fig. 2. Inspecting bridge network

```
PS C:\Users\whatanicedayiana\Documents\CoE197S\ME8_Containerization_and_Docker\5-volumes> docker run --rm -d --name dummy dianavsd25/ping:1.0
154a7411e0388cdf3ab7d1edac4c881d23329691a7e6c59b6b22fa33b47b91e2
PS C:\Users\whatanicedayiana\Documents\CoE197S\ME8_Containerization_and_Docker\5-volumes> docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "3f805bbb14928370c76ef1d254df6b1ac0cfb2623c4db3ced9571ec208371d6c",
        "Created": "2021-06-11T19:32:15.6929291Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "154a7411e0388cdf3ab7d1edac4c881d23329691a7e6c59b6b22fa33b47b91e2": {
                "Name": "dummy",
                "EndpointID": "617bb298c7a7f44b7a088b83d3f234b32990fc62f5d1aa12e906395545d81b0f",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
```

Fig. 3. Starting a *ping* container and inspecting bridge network again

```
PS C:\Users\whatanicedayiana\Documents\CoE197S\ME8_Containerization_and_Docker\5-volumes> docker run --rm -d -e PING_TARGET=172.17.0.2 --name pinger dia
navsd25/ping:1.0
ae97ec62dc6e449a24e8a7c2afd0816cdb480f25b7301ae3aaef51c0acc7c416
PS C:\Users\whatanicedayiana\Documents\CoE197S\ME8_Containerization_and_Docker\5-volumes> docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED         STATUS         PORTS     NAMES
ae97ec62dc6e   dianavsd25/ping:1.0      "sh -c 'ping $PING_T…"   7 seconds ago   Up 5 seconds             pinger
154a7411e038   dianavsd25/ping:1.0      "sh -c 'ping $PING_T…"   3 minutes ago   Up 3 minutes             dummy
PS C:\Users\whatanicedayiana\Documents\CoE197S\ME8_Containerization_and_Docker\5-volumes> docker logs pinger
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.045 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.079 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.076 ms
64 bytes from 172.17.0.2: icmp_seq=4 ttl=64 time=0.080 ms
64 bytes from 172.17.0.2: icmp_seq=5 ttl=64 time=0.080 ms
64 bytes from 172.17.0.2: icmp_seq=6 ttl=64 time=0.051 ms
64 bytes from 172.17.0.2: icmp_seq=7 ttl=64 time=0.080 ms
64 bytes from 172.17.0.2: icmp_seq=8 ttl=64 time=0.027 ms
64 bytes from 172.17.0.2: icmp_seq=9 ttl=64 time=0.241 ms
64 bytes from 172.17.0.2: icmp_seq=10 ttl=64 time=0.075 ms
64 bytes from 172.17.0.2: icmp_seq=11 ttl=64 time=0.028 ms
64 bytes from 172.17.0.2: icmp_seq=12 ttl=64 time=0.079 ms
64 bytes from 172.17.0.2: icmp_seq=13 ttl=64 time=0.041 ms
64 bytes from 172.17.0.2: icmp_seq=14 ttl=64 time=0.081 ms
64 bytes from 172.17.0.2: icmp_seq=15 ttl=64 time=0.075 ms
```

Fig. 4. Adding another *ping* container, and inspecting logs for it

```
PS C:\Users\whatanicedayiana\Documents\CoE197S\ME8_Containerization_and_Docker\5-volumes> docker run --rm -d -e PING_TARGET=dummy --name pinger dianavsd
25/ping:1.0
b0be859d1070846107a037076875198730679 9e270b15f07916699acbad2c1ac
PS C:\Users\whatanicedayiana\Documents\CoE197S\ME8_Containerization_and_Docker\5-volumes> docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED         STATUS         PORTS     NAMES
154a7411e038   dianavsd25/ping:1.0      "sh -c 'ping $PING_T…"   7 minutes ago   Up 7 minutes             dummy
PS C:\Users\whatanicedayiana\Documents\CoE197S\ME8_Containerization_and_Docker\5-volumes>
```

Fig. 5. Running *ping* with *dummy* as target causes error (host name couldn't be resolved), container exit and auto remove

3. Managing custom networks

```
PS C:\Users\whatanicedayiana\Documents\CoE197S\ME8_Containerization_and_Docker\5-volumes> docker network create skynet
44164c998d16e9f05f56a1cc771f571be3521f2f9dca5c95ce5f042b13bb9271
PS C:\Users\whatanicedayiana\Documents\CoE197S\ME8_Containerization_and_Docker\5-volumes> docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
3f805bbb1492   bridge    bridge    local
736823f9ad72   host      host      local
dcdc3c866b69   none      null      local
44164c998d16   skynet    bridge    local
PS C:\Users\whatanicedayiana\Documents\CoE197S\ME8_Containerization_and_Docker\5-volumes> docker network inspect skynet
[
    {
        "Name": "skynet",
        "Id": "44164c998d16e9f05f56a1cc771f571be3521f2f9dca5c95ce5f042b13bb9271",
        "Created": "2021-06-11T23:06:54.8621781Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
PS C:\Users\whatanicedayiana\Documents\CoE197S\ME8_Containerization_and_Docker\5-volumes> docker network rm skynet
skynet
PS C:\Users\whatanicedayiana\Documents\CoE197S\ME8_Containerization_and_Docker\5-volumes> docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
3f805bbb1492   bridge    bridge    local
736823f9ad72   host      host      local
dcdc3c866b69   none      null      local
```

Fig. 6. Creating, inspecting and removing a custom network

4. Adding containers to a network

```
PS C:\Users\whatanicedayiana\Documents\CoE197S\ME8_Containerization_and_Docker> docker network create skynet
3cfed9fa4138bd7489d6a840d1e4d276c8490658751bec3b147f0419d84ae145
PS C:\Users\whatanicedayiana\Documents\CoE197S\ME8_Containerization_and_Docker> docker run --rm -d --network skynet --name dummy dianavsd25/ping:1.0
bdcfe61b1a32f6bbe312e1ffa52045f41236d2355013b9c5ef9c0054c8660501
```

Fig. 7. Assigning *ping* container to a network

```
PS C:\Users\whatanicedayiana\Documents\CoE197S\ME8_Containerization_and_Docker> docker run --rm -d --network skynet -e PING_TARGET=dummy --name pinger d
ianavsd25/ping:1.0
9d1a0d72f0b28fd5771ecf6a8d1071be1fb310ef1ce17f5bc2b113cf0931d2b9
PS C:\Users\whatanicedayiana\Documents\CoE197S\ME8_Containerization_and_Docker> docker logs pinger
PING dummy (172.19.0.2) 56(84) bytes of data.
64 bytes from dummy.skynet (172.19.0.2): icmp_seq=1 ttl=64 time=0.068 ms
64 bytes from dummy.skynet (172.19.0.2): icmp_seq=2 ttl=64 time=0.048 ms
64 bytes from dummy.skynet (172.19.0.2): icmp_seq=3 ttl=64 time=0.030 ms
64 bytes from dummy.skynet (172.19.0.2): icmp_seq=4 ttl=64 time=0.088 ms
64 bytes from dummy.skynet (172.19.0.2): icmp_seq=5 ttl=64 time=0.089 ms
64 bytes from dummy.skynet (172.19.0.2): icmp_seq=6 ttl=64 time=0.104 ms
64 bytes from dummy.skynet (172.19.0.2): icmp_seq=7 ttl=64 time=0.084 ms
```

Fig. 8. *pinger* targeting dummy *ping* container resulting to a successful host name resolve

5. Connecting between containers in a network

```
PS C:\Users\whatanicedayiana> docker images
REPOSITORY          TAG       IMAGE ID       CREATED       SIZE
dianavsd25/ping     1.0       e71e95fa2303   3 hours ago   139MB
ubuntu              16.04     9ff95a467e45   3 weeks ago   135MB
postgres            11.0      7a2907672aab   2 years ago   311MB
PS C:\Users\whatanicedayiana> docker run --rm -d --name widgetdb --network skynet -p 5432 postgres:11.0
e603fe35d4846cf47abefa477cdf6f796fe5ae1f2018f280316a1d9f32e117aa
PS C:\Users\whatanicedayiana> docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED          STATUS          PORTS                     NAMES
e603fe35d484   postgres:11.0   "docker-entrypoint.s…"   24 seconds ago   Up 23 seconds   0.0.0.0:50982->5432/tcp   widgetdb
PS C:\Users\whatanicedayiana> docker run --rm -d --name gadgetdb --network skynet -p 5432 postgres:11.0
4785ff9d65d1c65b9e36d31073a58e731b31acc9fe694f2247a67d86b04f73ee
PS C:\Users\whatanicedayiana> docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED          STATUS          PORTS                     NAMES
4785ff9d65d1   postgres:11.0   "docker-entrypoint.s…"   4 seconds ago    Up 3 seconds    0.0.0.0:50987->5432/tcp   gadgetdb
e603fe35d484   postgres:11.0   "docker-entrypoint.s…"   39 seconds ago   Up 38 seconds   0.0.0.0:50982->5432/tcp   widgetdb
```

Fig. 9. Setting up two *postgres* databases to connect to one another

*Note: I don't know why, but the latest version of *postgres* creates an error for me. It immediately terminates the container in just a few milliseconds after I enter the *run* command (so I experimented, and used a *postgres:11.0*, and it worked)

```
PS C:\Users\whatanicedayiana> docker exec -it widgetdb /bin/bash
root@e603fe35d484:/# psql -U postgres
psql (11.0 (Debian 11.0-1.pgdg90+2))
Type "help" for help.

postgres=# \q
root@e603fe35d484:/# psql -U postgres -h gadgetdb
psql (11.0 (Debian 11.0-1.pgdg90+2))
Type "help" for help.

postgres=# \q
root@e603fe35d484:/# exit
exit
PS C:\Users\whatanicedayiana> docker stop widgetdb gadgetdb
widgetdb
gadgetdb
```

Fig. 10. Starting a shell session in the widgetdb using *docker exec*, and connecting to the local and gadget database

6. Binding ports to the host

```
PS C:\Users\whatanicedayiana> docker run --rm -d --name widgetdb --network skynet -p 5432:5432 postgres:11.0
0d9b6fd1737b9b57c0773389f6abc0f5438d8237851cbb432624b42fc89fda56
PS C:\Users\whatanicedayiana> docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED         STATUS         PORTS                                       NAM
ES
0d9b6fd1737b   postgres:11.0   "docker-entrypoint.s…"   5 seconds ago   Up 4 seconds   0.0.0.0:5432->5432/tcp, :::5432->5432/tcp   wid
getdb
PS C:\Users\whatanicedayiana> psql -U postgres -h localhost
```

Fig. 11. Bind ports from container to a port on host machine; *psql* command would work if it its installed in my machine (but it is not)

*Note: psql can be ran/accessed through this container if I have it installed (but I don't have it installed)